

Internet-Based Image Retrieval Using End-to-End Trained Deep Distributions

Alexander Vakhitov Andrey Kuzmin Victor Lempitsky
Skolkovo Institute of Science and Technology
Moscow, Russia

{a.vakhitov, lempitsky}@skoltech.ru, andrey.kuzmin@skolkovotech.ru

Abstract

Internet image search engines have long been considered as a promising tool for handling open-vocabulary textual user queries to unannotated image datasets. However, systems that use this tool have to deal with multi-modal and noisy image sets returned by search engines, especially for polysemous queries. Generally, for many queries, only a small part of the returned sets can be relevant to the user intent.

In this work, we suggest an approach that explicitly accounts for the complex and noisy structure of the image sets returned by Internet image search engines. Similarly to a considerable number of previous image retrieval works, we train a deep convolutional network that maps images to high-dimensional descriptors. To model image sets obtained from the Internet, our approach then fits a simple probabilistic model that accounts for multi-modality and noise (e.g. a Gaussian mixture model) to the deep descriptors of the images in this set. Finally, the resulting distribution model can be used to search in the unannotated image dataset by evaluating likelihoods of individual images.

As our main contribution, we develop an end-to-end training procedure that tunes the parameters of a deep network using an annotated training set, while accounting for the distribution fitting and the subsequent matching. In the experiments, we show that such an end-to-end approach boosts the accuracy of the Internet-based image retrieval for hold-out concepts, as compared to retrieval systems that fit similar distribution models to pre-trained features and to simpler end-to-end trained baselines.

1. Introduction

To pick up a new visual concept, most modern computer vision systems require a set of images depicting this concept. Such an image set is usually mined from the World Wide Web using an Internet image search engine or comes from a website containing tagged images. Most notable and influential in this respect is the Image-Net project [7] that

has the goal of obtaining a “clean” image set for each of the 80,000+ visual *synsets* corresponding to English nouns. For each such noun, the image set is first obtained by querying the search engine and is then curated through crowd-sourced human labour. The second step (screening by humans) represents a significant burden. As a result, while the positive influence of the Image-Net project on the fields of computer vision and artificial intelligence has been enormous, the progress towards its initial goal has stalled at about one quarter (at the time of submission 21,841 synsets out of 80,000 have been indexed).

The use of *uncurated* image sets from Internet search engines can potentially enable computers to learn visual concepts automatically and without humans in the loop. Such capability is highly beneficial for intelligent systems, especially in certain scenarios, such as *open-vocabulary* image retrieval that allows users to formulate queries to image collections using arbitrary natural language queries. The use of uncurated image sets obtained from the web, however, is known to be challenging [12, 17], since despite the ever-improving performance of image search engines, the returned image sets still contain irrelevant images, since many natural language queries are inherently polysemous, and since many visual concepts often correspond to different visual aspects (e.g. outdoor and indoor views of a certain landmark building).

Here, we introduce an approach for open-vocabulary image retrieval based on uncurated image sets such as returned by Internet search engines. Our approach is adapted to the noisy and multi-modal structure of such image sets. Similarly to a number of recent image retrieval/search systems [22, 3, 1, 2, 11], our approach is based on a convolutional neural network that maps images to high-dimensional descriptors. As our system effectively uses image sets as queries, at query time it fits a parametric probabilistic distribution such as a Gaussian or a mixture of Gaussians to the deep descriptors of all images in the query set. Thus, the query is represented not by a single vector or a mere collection of vectors, but by a “fully-fledged” probabilistic distribution (Figure 1). This distribution can then be used to

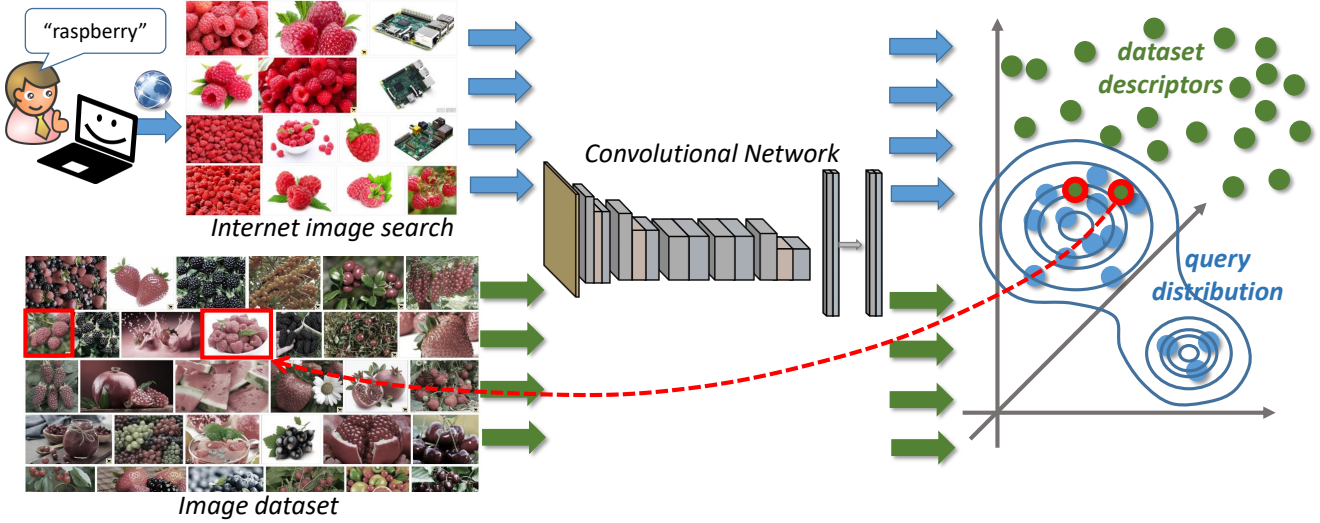


Figure 1. An overview of our system. The user intent is to find images of raspberry (berries) in an untagged image collection (bottom-left). The retrieval is based on the return of an Internet image search (top-left). Internet images and the images from the collection are mapped to descriptor vectors in the same space using a deep convolutional network. A distribution (e.g. a mixture of two Gaussians) is fitted to the descriptor vectors of the query set, and the likelihoods of the dataset descriptors are evaluated. Two images with high likelihoods (red) are returned as the response to the query. Our contribution is the end-to-end training procedure that optimizes the parameters of the convolutional network, so that the likelihoods of the relevant images w.r.t. fitted distributions is higher than the likelihood of irrelevant images.

e.g. find relevant images in an untagged dataset by evaluating the likelihoods of images.

Our key contribution is the end-to-end training procedure to the system that is described in the previous paragraph, based on a training set of images with known classes/concepts. Our training procedure thus tunes the weights of the underlying convolutional network to explicitly maximize the likelihoods of the relevant images and minimize the likelihood of irrelevant images under the fitted distributions on the training set. Towards this end, we derive and implement backpropagation through Gaussian mixture models (including single Gaussians). As we will show through the experiments on three different datasets, the ability to perform such an end-to-end training boosts the accuracy of image retrieval for hold-out classes not seen during training, as compared to baseline systems that have same or similar architecture but are not trained end-to-end.

Overall, we propose a general approach to on-the-fly learning of visual concepts using noisy image sets that combines probabilistic modeling and deep learning. Below, we briefly discuss relevant prior works in 2, detail our approach in 3, present results of experimental comparisons in 4 and conclude with a short discussion in 5.

2. Related work

On the early stage of development of visual object recognition, when image search relied mostly on textual information surrounding the image, the studies [9, 24] focused on

how to use visual content in order to improve ranking of the search results. Analogously to how document is represented by words in information retrieval, it was proposed to represent an image with 'visual words' describing its regions usually with SIFT descriptors. [9] applies techniques adopted from information retrieval to the problem of reranking, and [24] shows that the SVM classifier performs better in this problem.

The work [8] studied learning visual classification using Internet search engine output and adopted information retrieval methods to classify images with a model learned on search engine outputs. More related to our setting, [1] addresses large-scale image retrieval problem using image sets as queries and SIFT-based bag-of-words vectors as image features. Binary SVM learned on the query set with randomly sampled negatives, ranking using averaged query feature vectors (as defined in [21]) and averaging of the rankings for each individual query from the query set showed mean average precision (mAP) of 0.83-0.84 on the Oxford 105k dataset consisting of 5062 high resolution images of buildings and 100k other Flickr images serving as background. Various scenarios of image retrieval with single query image gave less accurate results.

The work [6] develops these methods further proposing cascades of classifiers for real-time on-the-fly object category retrieval in large image and video datasets. The follow-up work [5] considers three different types of queries for large-scale image retrieval and proposes to use bags of

visual words for object instance retrieval, SVM classifier trained on pre-learned CNN features for object category retrieval and an SVM classifier over face feature tracks for face retrieval in videos. Authors of [10] propose minimum description length-principle based data mining method on top of bags of visual words and show that it performs better than those proposed in [1] when the number of query images becomes bigger.

The work [19] describes learning a metric in the image feature domain in order to improve distance-based image classification and shows that the resulting metric generalizes well to the classes unseen during training. It also proposes a Nearest Class Mean (NCM) classifier as a distance to a mean of image class descriptors which we use as one of the baseline methods.

While we consider retrieval from unannotated test collection using a model built from a visual concept query, the work [12] proposes to generate image-to-text mapping for all the images in the test collection and then to process textual queries using this direct mapping. The approach was evaluated on objects from lab and kitchen environment, as the users were asked to formulate textual queries and the retrieval performance was measured. While this approach performs better than “blind” visual matching as in [1], it requires computationally intensive preprocessing which in fact is requesting a large database of images with detailed text annotations.

Finally, we mention a number of less related approaches. The work [17] describes a system for indexing user image collections with the help of Internet search and various data-sources such as maps to find locations. In [25], a robotic system is proposed that uses various Internet datasources including image search engine to learn how to perform certain complex activities. Their system learns object classes by training an SVM classifier on image sets resulting from multiple Internet search engine queries. The approach [18] uses Internet as a source of 3D models to learn 3D object classifiers in point clouds.

Apart from the works from the computer vision and the robotics communities discussed above, our approach is also related to certain directions pursued in the information retrieval and multimedia search communities. These include multiple query retrieval systems (e.g. [1]) and a large body of query reranking approaches, some of which use discriminative learning [13, 27].

3. Method

We now discuss the details of our system. At test time, the system takes a query set of images corresponding to a visual concept of interest, obtained using the Internet search engine and uses it to find images with this concept in the hold-out (“test”) collection. More precisely, the system ranks the test images according to their relevance to the vi-

sual concept (as in e.g. [1]).

To do the ranking, the system first computes the descriptor $d(I) = F(I; w)$ for every image I including images from the query set and from the test set using a convolutional neural network $F(\cdot; w)$ with parameters w that maps input images to the K -dimensional descriptor space. This computation happens on the fly for the query set $Q = \{I_1^q, I_2^q, \dots, I_{N_q}^q\}$ (which in our experiments has upto 100 images) and is performed offline for the test collection. When processing the query, the system fits a parametric generative distribution $p(\theta, \cdot)$ to the descriptor set of query images, where the process of fitting finds the optimal parameters of the distribution $\Theta(Q; w)$ using the maximum-likelihood principle:

$$\Theta(Q; w) = \operatorname{argmax}_{\theta} \sum_{j=1}^{N_q} L(\theta, F(I_j^q; w)) , \quad (1)$$

and where $L(\theta, F(I_j^q; w))$ is the log-likelihood of the parametric model evaluated for the descriptor of the j th image in the query. The system then evaluates the likelihood of the descriptors $d(I)$ for every test image I in the hold-out set. The produced likelihoods are then used for ranking.

3.1. Training process overview

Our training procedure is supervised. In particular, it assumes the availability of N_c visual concepts and of the training set of images with known relevance to those concepts. Importantly, the concepts used during training can be different from the concepts that our system can handle after deployment (in our experiments these two sets of concepts are disjoint). For each concept c at training time, we thus assume the availability of the query image set Q_c returned for this concept by the Internet search engine. The training process then seeks to minimize the following discriminative loss function:

$$\mathcal{L}(w) = \sum_c \mathbb{P}_{\substack{I_+ \in M_+(c) \\ I_- \in M_-(c)}} [L(\Theta(Q_c; w), F(I_+; w)) > L(\Theta(Q_c; w), F(I_-; w))] , \quad (2)$$

where the sum is taken over the training concepts c , $M_+(c)$ and $M_-(c)$ denote the sets of training images that are correspondingly relevant and irrelevant to the concept c and $\mathbb{P}_{\substack{I_+ \in M_+(c) \\ I_- \in M_-(c)}}(\cdot)$ denotes event probability while the images from the training sets are chosen at random. Furthermore, $\Theta(Q_c; w)$ are the parameters of the probabilistic model fitted to the set of deep descriptors as described by (1). Overall, despite complex notation, the loss (2) measures an intuitive quantity, which is the probability of the event that under the fitted parametric model, the image irrelevant to a concept will have a lower likelihood than a relevant image.

Optimizing the loss function over w makes the deep descriptors produced by the convolutional network more suitable for the retrieval using fitted distributions at test time.

The minimization of the loss (2) is done using stochastic gradient descent and backpropagation. To implement it we have to overcome two non-trivial hurdles. First, we need to find a way to compute the probability within (2) w.r.t. the likelihoods in a (piecewise)-differentiable and efficient manner. We further need to implement the backpropagation process through the distribution fitting operation (1), which involves finding the partial derivatives of the likelihoods w.r.t. the query image descriptors. Below, we discuss these two key operations (probability estimation and backpropagation through model fitting) starting from the latter.

3.2. Backpropagation through distribution fitting

As our parametric distributions in the descriptor space, we consider Gaussian mixture models (GMMs) with m components ($m \in \{1, 2, 3, 4\}$ in our experiments), all with diagonal covariance matrices, whereas the likelihood function is defined as a log-probability of such mixtures. The model parameters θ thus consist of the means denoted as $\{\mu_t\}_{t=1}^m$, the vectors $\{\phi_t\}_{t=1}^m$ parameterizing covariance matrices as $\Sigma_t = \text{diag}(\phi_t)$, and weights $\{v_t\}_{t=1}^m$ such that $\sum_{t=1}^m v_t = 1$.

To enable end-to-end learning of the deep image representation, we need to be able not only to compute the model parameters, which can be done in a standard way using the EM algorithm, but also to backpropagate through $L(\Theta(Q; w), x)$. We note that for $m = 1$ a model reduces to the Gaussian distribution, and there are closed-form expressions for its parameters, which can be analytically differentiated.

In the case of $m > 1$, general GMM fitting procedure outputs a model parameter vector $\theta_* = \Theta(Q; w)$ which maximizes log-likelihood function for the query set as defined in (1). We are then interested in the computation of $\nabla_{d_i} \theta_*$ that characterize the dependence of the optimal parameters θ_* on the query set descriptors $d_i = F(I_i^q, w)$.

We find these derivatives using the method of Lagrange multipliers (to handle the equality $\sum_{t=1}^m v_t = 1$) that allows to write for the optimal $\theta_* = \{\mu_{*,t}, \phi_{*,t}, v_{*,t}\}$ and the optimal lagrange multiplier λ_* :

$$\begin{aligned} \nabla_{\theta} \left\{ \sum_{j=1}^{N_q} L(\theta_*, d_j) + \lambda_* b(\theta_*) \right\} &= \mathbf{0}, \\ b(\theta_*) &= 0, \end{aligned} \quad (3)$$

where $b(\theta) = \sum_{i=1}^m v_i - 1$ is the constraint function ensuring that the weights of the mixture components sum to one.

To compute the required derivatives, we note that the optimal θ_* , λ_* are implicitly defined by (3), and thus the gradients of the left hand sides in (3) w.r.t. d_i are zero at θ_* ,

λ_* . This observation can be written down as the following system:

$$\begin{aligned} \nabla_{\theta, \theta} \left\{ \sum_{j=1}^{N_q} L(\theta_*, d_j) + \lambda_* b(\theta_*) \right\} \nabla_{d_i} \theta_* + \\ \nabla_{\theta} b(\theta_*) \nabla_{d_i} \lambda_* + \nabla_{\theta, d_i} L(\theta_*, d_i) &= \mathbf{0}, \\ \nabla_{\theta} b(\theta_*) \nabla_{d_i} \theta_* &= \mathbf{0}. \end{aligned} \quad (4)$$

The coefficients of this system (4) are the first and second derivatives of the functions $L(\theta_*, d_i)$ and $b(\theta_*)$, which we can compute because we know θ_* . The system (4) is linear w.r.t. the unknown matrix $\nabla_{d_i} \theta_*$ and vector $\nabla_{d_i} \lambda_*$ and has $(\dim(\theta_*) + 1) \times \dim(d_i)$ equations. After finding $\nabla_{d_i} \theta_*$, $\nabla_{d_i} \lambda_*$ we are able to compute the likelihood gradient w.r.t. the query descriptors using a chain rule by multiplying $\nabla_{d_i} L(\theta_*(d_i), d_i) = \nabla_{\theta} L(\theta_*, x) \nabla_{d_i} \theta_* + \nabla_{d_i} L(\theta_*, d_i)$.

To sum up, backpropagation operation through the distribution fitting layer involves solving the system of linear equations (4).

3.3. Estimating the expectation

We now come back to the task of estimating the probability for some relevant image to have a higher likelihood under the fitted model compared to some irrelevant image. To accomplish this, we use the modification of the histogram loss estimator recently proposed in [26]. Given the sets M_+ and M_- of relevant and irrelevant images, denote their deep descriptors as $\{d_{+,i}\}_{i=1}^{N_+}$ and $\{d_{-,i}\}_{i=1}^{N_-}$. Furthermore, let their likelihoods under the fitted distribution with the optimal parameters be $\mathbf{l}_+ = \{L(\theta_*, d_{+,1}), L(\theta_*, d_{+,2}), \dots, L(\theta_*, d_{+,N_+})\}$, $\mathbf{l}_- = \{L(\theta_*, d_{-,1}), L(\theta_*, d_{-,2}), \dots, L(\theta_*, d_{-,N_-})\}$.

The idea of the histogram loss [26] is to accumulate these two sets of numbers into histograms, and then estimate the required probability using these histograms. Here, to compute the histograms, we fix the triangular kernel density estimator $K(z, \omega)$ that for the argument z and the width parameter ω is defined as:

$$K(z, \omega) = \max\left\{1 - \frac{2|z|}{\omega}, 0\right\}. \quad (5)$$

We choose l_{\min} and l_{\max} to be the lower and the upper bounds of the numbers in the union of \mathbf{l}_+ and \mathbf{l}_- , and further accumulate the two histograms \mathbf{h}_+ and \mathbf{h}_- spanning the range from l_{\min} to l_{\max} having B bins each and corresponding to the sets \mathbf{l}_+ and \mathbf{l}_- respectively. As discussed in [26], the entries of the histograms \mathbf{h}_+ and \mathbf{h}_- depend in a differentiable manner on the entries of \mathbf{l}_+ and \mathbf{l}_- .

Given the two histograms, the desired probability can be approximated as:

$$\mathbf{P}_{l_+ \in \mathbf{l}_+ \atop l_- \in \mathbf{l}_-} [l_+ < l_-] \approx \sum_{i=1}^B h_{-,i} \sum_{j=1}^i h_{+,j},$$

where $h_{-,i}$ and $h_{+,j}$ denote the entries of the histograms.

3.4. Recap of the training process

The learning is driven by the minimization of the loss (2). This is accomplished by iterating through the training concepts, considering for each concept the composite batch that includes a sub-batch with the query images, a sub-batch with relevant images, and a sub-batch with irrelevant images. The three sub-batches are forward-passed through the convolutional network, whereas the sub-batch with the query images is subsequently passed through the distribution fitting layer. Once the descriptors for relevant and irrelevant sub-batches, as well as the probabilistic distribution are estimated, the histogram loss (3.3) is computed.

The backpropagation incurs first the backpropagation through the loss estimator (as in [26]), followed by the backpropagation through the distribution fitting layer (3.2), and finally the backpropagation of the gradients corresponding to all three sub-batches through the convolutional network, followed by the update of the network parameters.

4. Experiments

Below we provide the experimental evaluation of our system (for a different number of Gaussians within mixtures; with and without training). We also compare the performance to a number of baselines.

4.1. Protocols

We evaluate the methods for three different datasets. In each case, we split classes into training, validation, and testing. We form the training set out of the training classes, and we train the methods (including ours) on such classes. The methods are compared on the test set. During test, we use the mean average precision (mAP) as the accuracy metric. The meta parameters for all methods are tuned on the validation set.

Google image search is used to obtain the query sets at all stages (the API for the search engine typically returns 90-100 images). We use class names provided with the datasets to define text queries to the engine. No textual augmentations or search modifiers have been used. Since some of the considered datasets have overlaps with the search engine output, before running the experiments, we identified potential near-duplicates between the Google search results and the datasets using deep descriptors from non-finetuned convolutional network (AlexNet). We then manually checked the potential pairs and removed the true near duplicates from consideration.

4.2. Implementation details

We use Caffe [14] framework to work with convolutional networks. All our experiments are based on the AlexNet architecture [16] (Caffe version). We note that more modern deep convolutional architectures could be used in place of

AlexNet, however such substitution is likely to benefit all methods equally.

We use the 1000-dimensional features that are produced by AlexNet for the baselines that do not use fine-tuning/end-to-end learning (for the tasks we consider, these features performed optimally or close to optimally compared to other layers). When performing end-to-end learning, we replace the last layer with a smaller one that outputs 128-dimensional features. We perform L_2 -normalization of the descriptors at the end of the network. In all experiments, we initialize the network weights using the AlexNet provided with Caffe, while the last fully-connected layer for the end-to-end trained architectures is initialized randomly.

The distribution fitting layer and the loss layer have been implemented using Theano [4]. For back-propagation and learning we use the Caffe implementation of the ADAM algorithm [15] with momentum 0.9. We choose the learning rate and the termination moment using validation sets.

4.3. Baselines

Each baseline defines the likelihood function L_{\cdot} . The following baselines are considered:

- The mean-based system (AVG), which ranks the images in the test set based on the scalar product between their descriptors and the mean \bar{d} of the descriptors d_i of the query set:

$$\bar{d} = \frac{1}{N_q} \sum_{i=1}^{N_q} d_i, \quad L_{AVG}(x, \bar{d}) = \bar{d}^T x. \quad (6)$$

- The nearest neighbor (NN) ranker that ranks images in the test set based on the maximum of their scalar product with the query set descriptors:

$$L_{NN}(x, \{d_i\}_{i=1}^{N_q}) = \max_i d_i^T x. \quad (7)$$

- The support vector machine (SVM) 1-vs-all classifier as in [1], where SVM is learned using the query images as positive class and $2|Q|$ randomly sampled images from other queries as negative class. If we denote the weight vector of the SVM as $u(\{d_i\}_{i=1}^{N_q})$, then the ranking (“pseudo-likelihood”) function can be defined as:

$$L_{SVM}(x, \{d_i\}_{i=1}^{N_q}) = x^T u(\{d_i\}_{i=1}^{N_q}). \quad (8)$$

We evaluate these baseline methods for the pre-learned descriptors. We also consider fine-tuning of the convolutional network for the mean(AVG) and the nearest neighbor(NN) ranking. In this case, we use exactly the same learning architecture as explained in the previous section, but plug the corresponding likelihood function



Figure 2. Retrieval comparison for the Oxford Flowers dataset (query = ‘silverbush’). Left: part of the query image set obtained from Google Search API using query ‘silverbush’. Right: the top ranked images provided by various methods, namely AVG (average), NN (nearest neighbor), Gauss (one Gaussian), GMM2 (mixture of two Gaussians), GMM3 (mixture of three Gaussians) and GMM4 (mixture of four Gaussians) using the fine-tuned descriptors, SVM and GMM3PL (mixture of three Gaussians) using pre-learned ‘fc8’ features.

$L_{NN}(x, \{d_i\}_{i=1}^{N_q})$ or $L_{AVG}(x, \bar{d})$ instead of a more sophisticated distribution-based likelihood into the computation of the histogram loss.

Alongside the baselines, we report the results of our system for different number of Gaussians in the mixtures ($m = 1, 2, 3, 4$). We also report the results obtained with our architecture, when applied without learning (fine-tuning) by simply fitting Gaussian mixtures to pre-trained deep features.

4.4. ImageNet

We now present the results for individual datasets. Our first experiment uses classes from the ImageNet dataset [7]. Since we use networks pretrained on the ILSVRC classes [23], we made sure that 1000 classes included into the ILSVRC set are excluded from our experiments. Generally, the uncured Google image search outputs for some of the ImageNet synsets are very noisy (e.g. c.f. the synset ‘sociologist’).

To perform the experiments, we selected random 509

random synsets for training, 99 synsets for validation and 91 synset for testing. The results for our methods and the baselines are shown in the fig. 4. The results demonstrate that end-to-end learning is able to improve the mAP of the baseline methods by 1-4 percent and of the proposed methods based on distribution fitting by 5-7 percent. Importantly, the gap between the model that fits a single Gaussian and the model that uses the mean vector is almost 2 percent. Using mixture models with two or three components improves the performance further.

4.5. RGBD Object

The amount of relevant images in the Internet search results in this case differs greatly from category to category. For the polysemous categories such as ‘apple’, the proposed generative models can give significant benefit (Fig. 3). The quantitative results are given in fig. 5. They demonstrate even greater improvements from the end-to-end learning (perhaps due to a relatively bigger domain gap between the ImageNet and this dataset). Such training improves the

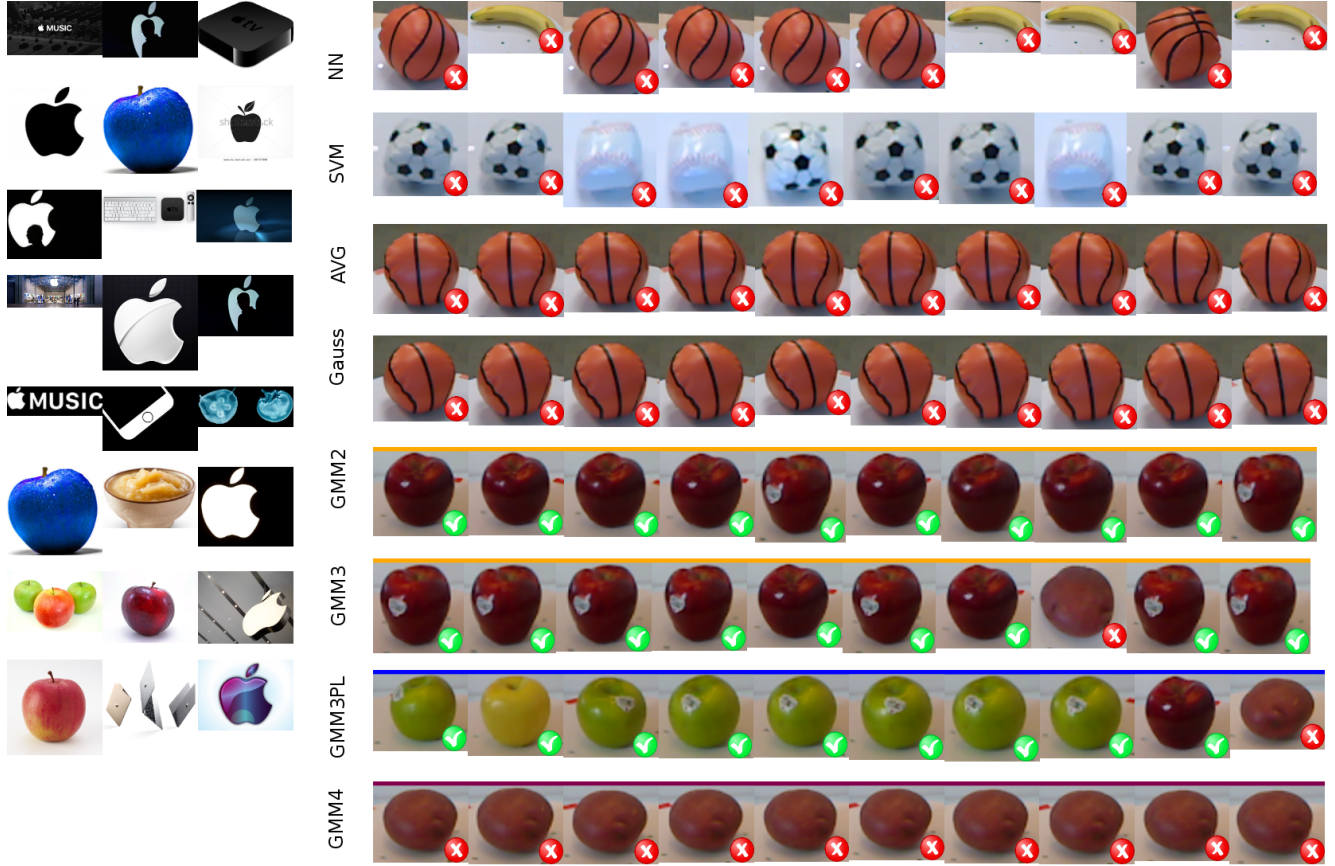


Figure 3. Retrieval comparison for the RGBD Object dataset (query = ‘apple’). Left: part of the query image set obtained from Google Search API using query ‘apple’. Right: the top ranked images provided by various methods, namely AVG (average), NN (nearest neighbor), Gauss (one Gaussian), GMM2 (mixture of two Gaussians), GMM3 (mixture of three Gaussians) and GMM4 (mixture of four Gaussians) using the fine-tuned descriptors, SVM and GMM3PL (mixture of three Gaussians) using the pre-learned ‘fc8’ features.

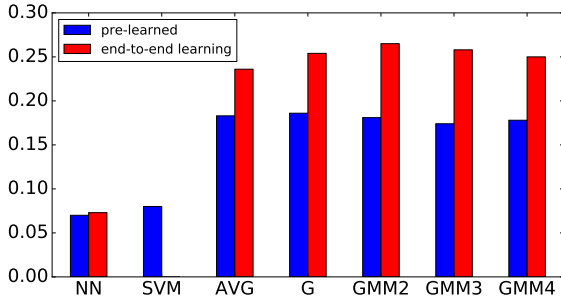


Figure 4. Mean average precisions for the **ImageNet** dataset (see text for more discussion). The gap between the mean-based and the Gaussian-based retrieval highlights the benefit of modeling uncertainty and noise in the Internet search engine output using probabilistic distributions.

map for the mean classifier by nine percent and for the proposed methods by 11-14 percent. The methods based on distribution fitting again perform better.

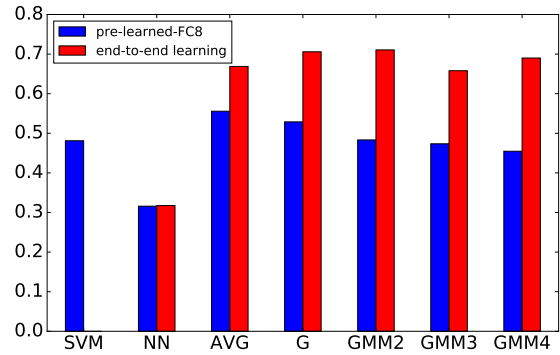


Figure 5. Mean average precisions for the **RGBD** dataset (see text for more discussion).

4.6. Oxford Flowers

We use the Oxford Flowers-102 dataset [20], consisting of images of 102 different UK flowers. The dataset was split into 80 categories for training and validation, and 22

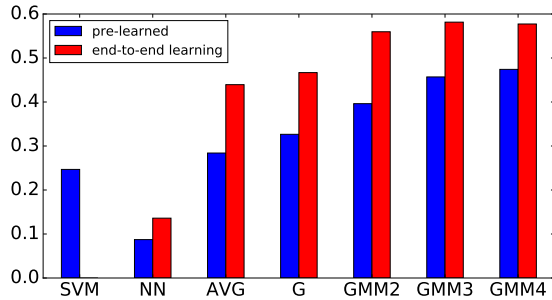


Figure 6. Mean average precisions for the **Oxford Flowers** dataset (see text for more discussion). In this case, the biggest boost from the end-to-end learning as well as the biggest gap between single Gaussian and multiple component Gaussian mixtures are observed.

for testing (see supplemental material). The results at fig. 6. The end-to-end learning procedure improves the mAP on 14 percent for the mean classifier, 16-19 percent for the proposed models. The fig. 2 shows an example of the polysemous query ('silverbush'), where the ability of the Gaussian mixture models to capture multi-modal distributions provides our approach a big advantage.

5. Conclusion

In this work, we have introduced a new architecture for image retrieval based on noisy image sets obtained from Internet image search engines. Our architecture combines deep networks with the probabilistic distribution fitting, and can be trained end-to-end on a separate annotated training set.

Through extensive comparisons with various baselines, we have shown that such end-to-end training results in an architecture that generalizes across classes and can handle the challenges of Internet-based visual concept modeling better than baseline models.

References

- [1] R. Arandjelovic and A. Zisserman. Multiple queries for large scale specific object retrieval. In *BMVC*, pages 1–11, 2012. 1, 2, 3, 5
- [2] A. Babenko and V. S. Lempitsky. Aggregating deep convolutional features for image retrieval. In *International Conference on Computer Vision - ICCV*, 2015. 1
- [3] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision - ECCV*, pages 584–599, 2014. 1
- [4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010. 5
- [5] K. Chatfield, R. Arandjelović, O. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *International journal of multimedia information retrieval*, 4(2):75–93, 2015. 2
- [6] K. Chatfield and A. Zisserman. Visor: Towards on-the-fly large-scale object category retrieval. In *Asian Conference on Computer Vision*, pages 432–446. Springer, 2012. 2
- [7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 1, 6
- [8] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google’s image search. In *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, volume 2, pages 1816–1823. IEEE, 2005. 2
- [9] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In *European Conference on Computer Vision*, pages 242–256. Springer, 2004. 2
- [10] B. Fernando and T. Tuytelaars. Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2544–2551, 2013. 3
- [11] G. T. Filip Radenovi and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision - ECCV*. 1
- [12] S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell. Open-vocabulary object retrieval. In *Robotics: science and systems*, volume 2, page 6, 2014. 1, 3
- [13] V. Jain and M. Varma. Learning to re-rank: query-dependent image re-ranking using click data. In *Proceedings of the 20th international conference on World wide web*, pages 277–286. ACM, 2011. 3
- [14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 5
- [15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 5
- [17] N. Kumar and S. Seitz. Photo recall: Using the internet to label your photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 771–778, 2014. 1, 3
- [18] K. Lai and D. Fox. Object recognition in 3d point clouds using web data and domain adaptation. *I. J. Robotic Res.*, 29(8):1019–1037, 2010. 3
- [19] T. Mensink, J. Verbeek, F. Perronnin, and G. Csorika. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013. 3

- [20] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE, 2008. 7
- [21] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 2
- [22] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Visual instance retrieval with deep convolutional networks. *CoRR*, abs/1412.6574, 2014. 1
- [23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 6
- [24] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. *IEEE transactions on pattern analysis and machine intelligence*, 33(4):754–766, 2011. 2
- [25] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *Robotics Automation Magazine, IEEE*, 18(2):58–68, June 2011. 3
- [26] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 4, 5
- [27] L. Yang and A. Hanjalic. Supervised reranking for web image search. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 183–192. ACM, 2010. 3